SPECIFICATION AMENDMENTS:

Kindly amend the last paragraph of page 1 extending over to pages 2 and 3 as follows --

With the release of the Lightweight Directory Access Protocol (LDAP) version 3 in 1997 [M. Wahl, T. Howes, and S. Kille. Lightweight directory access protocol (v3). RFC 2251, December 1997.], the popularity of directories to store information about users, networks, etc. has been steadily increasing. Even companies like Netscape or Microsoft offer LDAP support in their browsers and operating systems, making directory services a viable alternative to more traditional database systems for the storage and efficient retrieval of information. At the same time, the Internet community has been moving away from static HTML to describe information on the web, towards more dynamic and easily configurable options that allow the decoupling of content, usually represented in form of XML data [Tim Bray, Jean Paoli, C.M. Sperberg-McQueen, and Eve Maler. Extensible markup language (XML) 1.0 (second edition). ~~http://~~www.w3.org/TR/2000/REC-XML-20001006, October 2000.], and format, usually represented as CSS data [Hakon Wium Lie and Bert Bos. Cascading style sheets, level 1. ~~http://~~www.w3.org/TR/REC-CSS1, January 1999.]. This transition has lead to an increase in the involvement of the database community in issues related to semi-structured databases [H. V. Jagadish, Laks V. S. Lakshmanan, Tova Milo, Divesh Srivastava, and Dimitra Vista. Querying network directories. In SIGMOD 1999, Proceedings ACM SIGMOD International Conference on Management of Data, June 1-3, 1999, Philadephia, Pennsylvania, USA, pages 133-144. ACM Press, 1999], a

reevaluation of semi-structured data models, and even to the creation of models and mechanisms to efficiently represent and process semi-structured data in relational database systems [Albrecht Schmidt, Martin L. Kersten, Menzo Windhouwer, and Florian Waas. Efficient relational storage and retrieval of XML documents. In Proceedings of the Third International Workshop on the Web and Databases, pages 47-52, Dallas, Texas, May 2000. Khaled Yagoub, Daniela Florescu, Valerie Issarny, and Patrick Valduriez. Caching strategies for data-intensive web sites. In Proceedings of the 26th VLDB Conference, Cairo, Egypt, 2000]. These relational systems have, nevertheless, limitations due to differences in the representation and query model [Jayavel Shanmugasundaram, Kristin Tufte, Gang He, Chun Zhang, David DeWitt, and Jeffrey Naughton. Relational databases for querying XML documents: Limitations and opportunities. In Proceedings of the 25th VLDB Conference, Edinburgh, Scotland, 1999.] that support the need for XML processing systems closer to its semi-structured nature. --.

Please amend the last paragraph of page 13 and the first paragraph of page 14 as follows --

Figure 1 depicts a group of four browsers (B) retrieving information from three different information systems (IS) through the inventive ~~Inventive~~ system, composed, in this particular example, of two levels of proxy caching servers that exchange information ~~both,~~ both at the inter-level and the intra-level domains.

The configuration of the individual proxy cache nodes in the inventive ~~Inventive~~ system, as well as their belonging to a particular level in the

hierarchy are managed by the underlying Plexor system, which also provides facilities for scalability, replication, fault-tolerance and the transparent addition or removal of nodes in the system.
Internally, each one of the proxy cache nodes in figure 1 is made up of the following components, schematically represented in figure 2:

- Proxy Web Server (part of the Apache server)
- Query Engine
- Distributed Cache Engine
- XML Parser
- XMLDAP Cache (OpenLDAP Server)
- Traditional Cache (also part of the Apache server) --.

Please amend the last paragraph of page 14 as follows --

A modified version of the popular Apache Server [Apache Group. Apache web server. ~~http://~~www.apache.org/.] is heavily used in the inventive system to provide the functionality of a caching proxy server that forwards user requests as needed, either to the appropriate node in the hierarchy or directly to the information system after the corresponding processing has been done by the node. --.

Kindly amend the first paragraph of page 15 as follows --

As it is obvious from figure 2, the proxy engine serves as a front-end that collects queries in the form of URL requests, and transmits the result (output) back to the client. Whether or not the node has the necessary information in either one of its caches to provide the answer

without incurring ~~in the~~ the overhead of contacting other nodes is determined by the Query Engine and the Distributed Cache Engine. --.

Please amend the second paragraph of page 15 as follows --

If, as ~~it may~~ may be the case, the node does not have the necessary information to answer the request by itself, the query is forwarded to the next level in the hierarchy, and the answer cached in the node as it is sent back to the client. Whether the object should be cached in the Traditional Cache or in the XMLDAP Cache is determined by the Type Checker. --.

Kindly amend the last paragraph of page 15 extending to page 16 as follows --

The Query Engine is a custom-made component designed to process, decompose and normalize XPath [James Clark and Steve DeRose. XML path language (XPath) version 1.0. ~~http://~~www.w3c.org/tr/xpath, November 1999] queries into LDAP constructs that the XMLDAP Cache can understand. Any other type of queries, like URL requests that do not contain XPath expressions, are forwarded to the Distributed Cache Engine to determine whether or not the system as a whole could process the request. The decomposition and normalization of queries plays an extremely important role in the cache answerability problem, and more importantly, in the efficient processing and delegation of queries and subqueries, as will be seen below. --.

Please amend the first full paragraph on page 16 as follows --

In order to speed up the querying process, the Query Engine works in close ~~tight~~ cooperation with the Distributed Cache Engine, whose only purpose so far, is the management of a distributed index of the contents of both, the XMLDAP Cache and the Traditional Cache from all nodes in the system. In the future, the Distributed Cache Engine will also be responsible for the efficient integration and processing of Document Type Definitions (DTDs) to allow for the correct characterization and utilization of semantically related cache entries with different syntax. --.

Kindly amend the last paragraph of page 17 as follows --

As an additional feature, the XML Parser also handles the conversion of an LDAP tree structure to XML, allowing for on-the-fly generation of well-formed XML documents from partial documents cached at a previous time. The XMLDAP Cache is based on the last available version of the OpenLDAP server [OpenLDAP Group. OpenLDAP server. ~~http://~~www.openldap.org/.] modified to provide the higher performance requirements typical of a caching system. The purpose of this component is to serve as a specialized cache for XML documents, and therefore, the standard configuration has been changed to fulfill the requirements imposed by the storage of generic XML documents, as will be seen below. --.

Please amend the last paragraph of page 22 as follows --

The XMLElement class differs from the XMLAttribute class in that the former requires the presence of an order attribute, whereas the latter does not even allow it. This attribute is used to encode the relative ordering of a particular node in the document hierarchy which, as will be seen later, is crucial for the correct implementation of queries like next-node, previous-node, next-sibling, etc., as defined in the XPath standard [James Clark and Steve DeRose. XML path language (XPath) version 1.0. ~~http://~~www.w3c.org/tr/xpath, November 1999]. --.

Kindly amend the first full paragraph of page 26 as follows --

The next four attributes, context, scope, xpathquery and result define a query or subquery in terms of the characteristics described in the XPath specification [James Clark and Steve DeRose. XML path language (XPath) version 1.0. ~~http://~~www.w3c.org/tr/xpath, November 1999]. The context attribute is a set of distinguished names defined as the result set of a (possibly non-existing) previous subquery. The contents of the result attribute is the set of distinguished names that contain the LDAP nodes resulting from applying the query stored in the xpathquery attribute under the scope defined in the scope attribute on the context of the query. By means of these four attributes, the inventive caching system is able to provide support for subquery rewriting, remote query processing, cache answerability, and other features that will be dealt with in detail below. --.

Please amend the first paragraph of page 29 extending to page 30 as follows --

The query model used by the invention is very close to the traditional LDAP query model described in the standard specification [M. Wahl, T. Howes, and S. Kille. Lightweight directory access protocol (v3). RFC 2251, December 1997.] and in other pieces of the literature [T. A. Howes, M. C. Smith, and G. S. Good. Understanding and Deploying LDAP Directory Services. Macmillan Network Architecture and Development. Macmillan Technical Publishing U.S.A., 1999. H. V. Jagadish, Laks V. S. Lakshmanan, Tova Milo, Divesh Srivastava, and Dimitra Vista. Querying network directories. In SIGMOD 1999, Proceedings ACM SIGMOD International Conference on Management of Data, June 1-3, 1999, Philadephia, Pennsylvania, USA, pages 133-144. ACM Press, 1999]. However, the inventive model differs from previous approaches in the inventive desire to limit to a minimum the number of changes to the standard LDAP model so that it can be deployed easily in existing systems, while at the same time providing full XPath support [James Clark and Steve DeRose. XML path language (XPath) version 1.0. ~~http://~~www.w3c.org/tr/xpath, November 1999]. Other papers [H. V. Jagadish, Laks V. S. Lakshmanan, Tova Milo, Divesh Srivastava, and Dimitra Vista. Querying network directories. In SIGMOD 1999, Proceedings ACM SIGMOD International Conference on Management of Data, June 1-3, 1999, Philadephia, Pennsylvania, USA, pages 133-144. ACM Press, 1999] provide extensions to the LDAP model that, although very interesting and valid, either go beyond the requirements of XPath, or need considerably more work than the inventive model in order to be deployed in current systems. --.

Please amend the last paragraph of page 31 extending to page 32 as follows --

As specified in [James Clark and Steve DeRose. XML path language (XPath) version 1.0. ~~http://~~www.w3c.org/tr/xpath, November 1999], the primary purpose of the XPath standard is to address parts of an XML document, usually represented in the form of a tree that contains element, attribute and text nodes. An XPath Query Q_X is formed by the concatenation of path expressions that perform walk-like operations on the document tree retrieving a set of nodes that conform to the requirements of the query. Each expression is joined with the next by means of the classical Unix path character '/'. --.

Kindly amend the first full paragraph of page 38 as follows --

In order to prove the feasibility ~~of The~~ of the invention as an efficient cache for XML, a series of experiments have been performed to determine the following characteristics of the inventive system: document storage overhead; average storage and retrieval time; and query execution performance improvement. Replacement policies of any kind have not been considered in the inventive system because the aim of the inventive experiments is to prove the feasibility of the invention as a caching mechanism for XML, and, for now, not to devise or propose new replacement policies. Preliminary experiments ~~that~~ involve more than one server and make use of the parallelization capabilities of XPath queries in the inventive system. --.

Please amend the last paragraph of page 39 extending to page 40 as follows --

The data files used in the inventive benchmarking experiments come from several sources: the Mondial database that contains geopolitical information about countries, organizations, geographical entities, etc.; XSLBench [Kevin Jones. XSLBench 1.4. ~~http://~~www.tfi-technology .corn/XML/xslbench.html, October 2000.], a performance benchmark of XSLT [James Clark. XSL transformations (XSLT) version 1.0. ~~http://~~www.w3.org/TR/xslt, November 1999.
] processors maintained by Kevin Jones; the ACM Sigmod Record Database [ACM. Sigmod record web edition. ~~http://~~www.acm.org/sigs/sigmod/record, December 2000.
] in XML form; and a database of "Great Books" maintained in WML [~~http://~~www.oasis-open.org/cover/wap-wml.html.] at JollyRoger [Jollyroger.com great books. ~~http: //~~jollyroger. nbci.com.], which serves as an example of the use of the invention with WAP technology [~~http://~~www.wapforum.org/.], since WML is nothing but a set of XML documents that conform to the WML Document Type Definition. --.

Kindly amend the last paragraph of page 40 as follows --

As can be seen in table 3, the overhead of the traditional cache is minimal, since only a header containing information about the caching time, expiration date, size of the file, etc. is stored with the file. Therefore, small files suffer from a greater relative overhead ~~as big than~~ big files. For the inventive internal representation, the storage requirements are about 2.8 times that of the original size of the file, with no direct correlation between size and relative overhead, as in the previous case. Although the inventive representation has greater overhead than the traditional cache representation, the additional

querying capabilities of the inventive system make it a reasonable tradeoff. --.

Please amend the first paragraph of page 41 as follows --

Table 4 contains data on the storage and retrieval times of the same files used for the storage requirement experiment. The storage operation involves loading an XML document into HLCaches by means of the XML2LDAP algorithm, after a query has been formulated, whereas the retrieval operation assumes that the document is already in the inventive system and needs to be reconstructed to be returned to the client. For this experiment, the absolute size of the file is not so important as the number of element and attribute nodes in the document. The storage and retrieval times have been measured as seen by the Apache server after performing the corresponding operations, and are always greater than the ones seen by the clients, since they start receiving data from the server before the whole operation is complete, which helps achieve ~~perceive~~ even better times from the client perspective. --.

Kindly amend the last paragraph of page 41 extending to page 42 as follows --

As depicted in table 4, the invention can process ~~almost~~ nearly 4700 store operations per second, which correspond to about 2700 XML nodes/second, ~~where~~ with each node ~~is either~~ being either an element or an attribute. The performance of storage operations is ~~so good in comparison to the~~ superior to that of retrieval operations because they

are performed asynchronously, meaning that the Apache server does not need to wait for the LDAP server to complete the operation before it sends the next one. On the other hand, retrieval operations are synchronous, since the order in which nodes are received is an important factor for the reconstruction of the original document from its individual nodes. Despite the relative performance disadvantage of read operations overall, the fact that clients start receiving the document as soon as the first bytes are generated, and that most documents are not as big as the ones used in the inventive experiments, imply that there is no noticeable overhead for read operations seen from the perspective of the client. --.

Please amend the last paragraph of page 42 as follows --

In the third set of experiments, one has tried to determine the relative performance gain of the inventive query mechanism with respect to similar XPath engines by the proper use of LDAP filters and translation mechanisms as detailed in section 4. For the following set of queries, a C-based implementation of an XPath engine [Daniel Veillard. ~~http://~~www.XMLsoft.org/.], developed for the Gnome project [GNOME.~~http://~~www.gnome.org/.] by Daniel Veillard, has been taken. At the time when the inventive experiments initiated, this engine was the only open source XPath implementation of which one was aware, written in C that could be compared to the inventive system (also written in C). The original version was modified to use the inventive system as a substitute for the DOM representation needed to perform XPath queries on a XML document.

--.

Kindly amend the last paragraph of page 43 extending to page 44 as follows --

In the world of caching technologies for the Internet, the two most prominent examples of hierarchical caching systems are Harvest [C. Mic Bowman, Peter B. Danzig, Darren R. Hardy, Udi Manber, Michael F. Schwartz, and Duane P. Wessels. Harvest: A scalable, customizable discovery and access system. Technical report, University of Colorado at Boulder, March 1994, Anawat Chankhunthod, Peter B. Danzig, Chuck Neerdaels, Michael F. Schwartz, and Kurt J. Worrell. A hierarchical internet object cache. In Proceedings for USENIX '96, 1996] and Squid [M. Hamilton, A. Rousskov, and D. Wessels. Cache digest specification - version 5. ~~http://~~squid.nlanr.net/CacheDigest/cache-digest-v5.txt,December 1998]. Although the inventive system also provides a hierarchical caching structure, it differs from both systems, first, in that it also provides querying facilities, acting more like materialized view layer of a semi-structured database than a traditional cache. Secondly, the inventive system can reconfigure its topology on-the-fly, allowing for optimization techniques and methods not available on Harvest or Squid. Finally, the inventive integration approach for arbitrary semi-structured documents is generic, as opposed to the wrapper technology used by Harvest to implement brokers for indexing specific data formats. --.

And kindly amend the last paragraph of page 44 extending to page 45 as follows --

The inventive approach is also similar to some active caching techniques, since it offers additional features and capabilities to non-traditional cacheable types, as opposed to HTML, image files, etc. However, the usual focus of active caching systems [Pei Cao, Jin Zhang, and Kevin Beach. Active cache: Caching dynamic contents on the Web. In Proceedings of the 1998 Middleware conference, September 1998. ~~http://~~www.cs.wisc.edu/-cao/papers/active-cache.html], lie on applications and applets either in Java or Javascript, whereas the inventive system focuses on semi-structured data and how to efficiently query it, and not take into account the more programmatic cacheable entities that populate the World Wide Web. --.